Amendments to the Specification

Please replace the paragraph that begins on Page 1, line 6 and carries over to Page 2, line 1 with the following marked-up replacement paragraph:

-- The present invention is related to U. S. Patent 6,757,708, titled ~~Patent _____, titled~~ "Caching Dynamic Content" (serial number 09/518,474), which was filed on 03/03/2000; U. S. Patent 6,505,200, titled "Application-Independent Data Synchronization Technique" (serial number 09/611,030), filed concurrently herewith; and U. S. Patent 6,665,867, titled "Self-Propagating Software Objects and Applications" (serial number 09/610,513), also filed concurrently herewith. These related inventions are all commonly assigned to International Business Machines Corporation (IBM), and are hereby incorporated herein by reference. --

Please replace the paragraph that begins on Page 6, line 11 and carries over to Page 7, line 10 with the following marked-up replacement paragraph:

-- Furthermore, there may be repeated requests for retrieval of the same information. If repetitively requested information tends to be somewhat static in nature, it is an inefficient waste of system resources to interact with the back-end system each time it is requested, only to retrieve the same result that was obtained with a prior request. In addition, an application program may generate updates to a back-end data store which are not time-critical. An example of this type of application is one that generates low-priority processing requests such as daily purchase orders, where it might not be necessary to process the orders immediately: rather, delayed execution could process the orders and send confirmation messages to the initiators. Many other examples of applications which generate updates that do not require immediate, real-time processing exist. For such

applications, it may be preferable for the updates to be accumulated over time and processed when the receiving computing system is lightly loaded, enabling the system's scarce resources to yield to higher-priority tasks in the interim. The prior art does not provide general solutions for optimizing resource utilizations in this manner. Instead, a developer must manually code logic to optimize resource usage, in view of the needs of a particular application, leading to complex (and therefore error-prone) programming requirements. The related U. S. Patent 6,757,708 titled ~~Patent titled~~ "Caching Dynamic Content" (serial number 09/518,474, referred to hereinafter as the "first related invention") defines a technique for caching objects (which may be JavaBeans) to avoid the system overhead of repetitive retrieval of information which has not changed. While the technique disclosed therein provides an efficient way to deal with read access to objects, it does not address write access. --

Please replace the paragraph on Page 27, lines 6 - 19 with the following marked-up replacement paragraph:

-- The cache manager caches RA objects in a data repository (i.e. a cache) which may be internal to the cache manager, or which is otherwise accessible thereto. When an application requests access to a cached bean, the access may be either RA or WA. If the request by a client application is for a RA object, this request is satisfied from the cache store without accessing the back-end data source, provided that a cached object exists which can satisfy the request. Each cached object preferably includes a cache policy, which is preferably specified by the developer as a method of each bean instance. (Alternatively, the cache policy may be specified on a per-class basis, with each bean instance inheriting the method of its associated class. Cache policy may be

specified as a class or instance variable, instead of as a method, provided that a method of the cache manager is properly adapted to processing the variable's value. This latter approach may be used, for example, if all cache policy values are to be interpreted as a time-of-day value.) If desired in a particular implementation of either or both aspects of the present invention, an administrative interface may be provided to enable the caching policy and/or update mode selection, to be described below, to be explicitly set (e.g. as configuration parameter values). --

Please replace the paragraph that begins on Page 30, line 17 and carries over to Page 31, line 13 with the following marked-up replacement paragraph:

-- A request to a WA object results in the back-end data source being updated in one of three possible modes: (1) a synchronous update; (2) an asynchronous update; or (3) a queued disconnected update, which may be referred to equivalently as a delayed update. Fig. 3B illustrates examples of the flows of this update process. As an application 335 executes, it requests some number of updates (shown as "351 ...") to a particular object, which in the example of Fig. 3B is HAO3 (element 360). If application 335 is operating in disconnected mode, then these updates cannot be processed against the back-end data source in real-time. Rather, the updates must be accumulated and applied to the back-end data source at some later time. In the branch office scenario, for example, the branch's daily work may be accumulated for transmission to the back-end enterprise system for processing after the branch closes for the day. Or, in the mobile computing scenario, the requests may be accumulated for subsequent transmission when the mobile ~~devices~~ device connects to a server. After transmission, the mobile device may then disconnect (to reduce connection costs, for example), and will receive the server's responses during some (arbitrarily-

timed) subsequent connection. This queued disconnected mode is preferably used for the branch office and mobile computing scenario, and may be used for other scenarios as well: application-specific criteria may be used to determine which update mode to use. –